**Министерство образования Республики Беларусь**

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

ОТЧЁТ

По лабораторной работе

На тему:

СОРТИРОВКИ

|  |  |  |
| --- | --- | --- |
| Выполнил  Студент гр. 751007 |  | Плескач К. С. |
| Проверил |  | Асс. С.В. Болтак |
|  |  | |
|

Минск, 2018

1. **Условие задачи**

Произвести анализ сортировок массивов в соответствии с вариантом, выданным преподавателем. Для чего отсортировать по возрастанию массивы целочисленных элементов различной размерности: 10 элементов, 100 элементов, 2000 элементов. Анализ произвести по числу сравнений и перестановок двух элементов. Исследования производить над массивами трех типов:

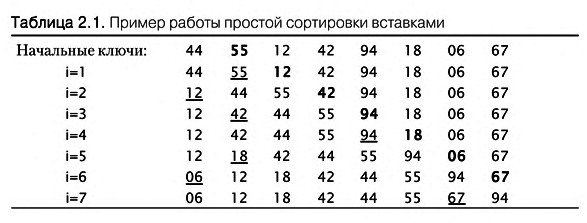
1. Массив, содержащий неотсортированные элементы;
2. Массив, содержащий отсортированные элементы;
3. Массив, содержащий элементы, отсортированные в обратном порядке.

Вариант задания (9): Бинарные вставки; Шейкерная сортировка, Quicksort.

1. **Описание сортировок**

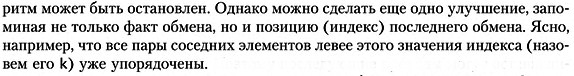
2.1. Бинарные вставки

По сравнению с простой сортировкой вставками для более быстрого нахождения позиции вставки в бинарных вставках используется алгоритм деления пополам, в котором проверяется середина последовательности-приемника и затем продолжаются деления пополам, пока не будет найдена точка вставки.



2.2. Шейкерная сортировка:

Это усовершенствованная пузырьковая сортировка.



2.3. Быстрая сортировка:

Является усовершенствованием метода сортировки обменами. Суть метода состоит в следующем:

1. Выбирается некоторый элемент Х из массива А, состоящего из n элементов (в качестве такого барьерного элемента может быть выбран центральный элемент массива).
2. Массив просматривается в прямом направлении (i=1,2,… и так далее до центрального элемента) с целью поиска в нём элемента А[i], не меньшего, чем Х.
3. Массив просматривается в обратном направлении (j=n,n-1,… и так далее до центрального элемента), ведётся поиск элемента А[j], не превосходящего Х.
4. Производится перестановка элементов А[i] и А[j] местами.
5. Шаги 1-4 повторяются, пока i<j.

В результате таких действий слева от элемента Х окажутся элементы меньшие или равные Х, а справа – элементы, большие Х. Пусть при этом элемент Х попадёт в позицию с номером k, тогда массив будет иметь вид

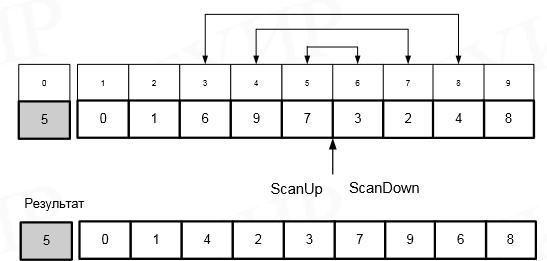
А[1], A[2], …, A[k-1], A[k], A[k+1], …, A[n].

Каждый из элементов А[1], A[2], …, A[k-1] меньше или равен A[k], а каждый из элементов A[k+1], …, A[n] больше A[k]. Отсюда можно сделать вывод, что элемент A[k] стоит на своём месте. Исходный массив при этом разделится на две независимые неотсортированные части.

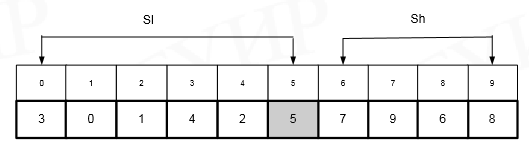
Для дальнейшей сортировки шаги 1-5 применяются для каждой из частей массива. И так до тех пор, пока в массиве не останутся подмассивы, состоящие из одного элемента, т.е. пока массив не будет отсортирован полностью.

Пример выполнения сортировки массива:

Пусть дан массив А={0, 1, 6, 9, 5, 7, 3, 2, 4, 8}







1. Далее алгоритм повторяется для образованных подмассивов.
2. **Схемы алгоритмов сортировок**
3. Бинарные вставки:



1. Шейкерная сортировка:



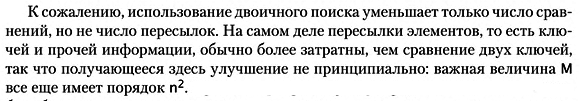
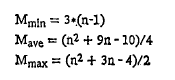
1. Быстрая сортировка:



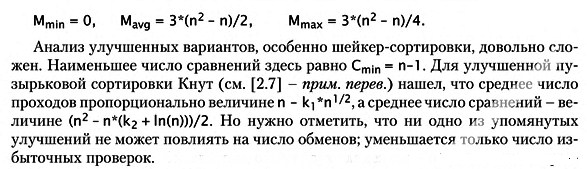
1. **Анализ сортировок**
2. Бинарные вставки:

Методы «бинарного поиска» указывают, куда вставлять j-й элемент после приблизительно lnj соответствующим образом выбранных сравнений. Например, если вставляется 64-я запись, можно сначала сравнить ключ K64 с K32; затем, если он меньше, сравниваем его с K16, если больше – с K48 и т. д., так что место для R64 будет найдено после ,всего лишь, шести сравнений. Общее число сравнений для N вставляемых элементов равно приблизительно NlnN, что существенно лучше, чем N2/4.

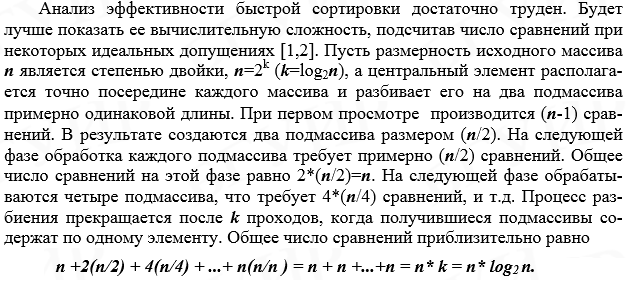
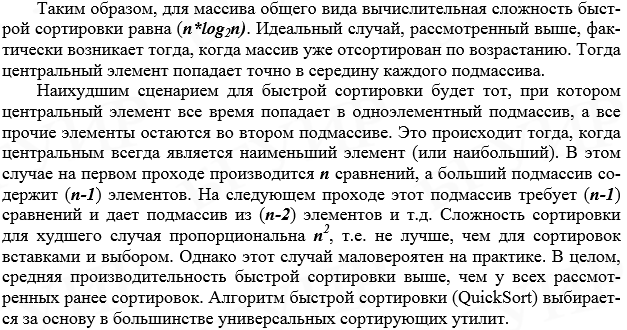
Неприятность состоит в том, что бинарные вставки решают задачу только на половину: после того, как мы нашли, куда вставлять запись Rj, все равно нужно подвинуть примерно j/2 ранее отсортированных записей, чтобы освободить место для Rj, так что общее время работы остается, по существу, пропорциональным N^2. Минимальное количество перестановок = n-1. Среднее количество перестановок:



1. Шейкерная сортировка:

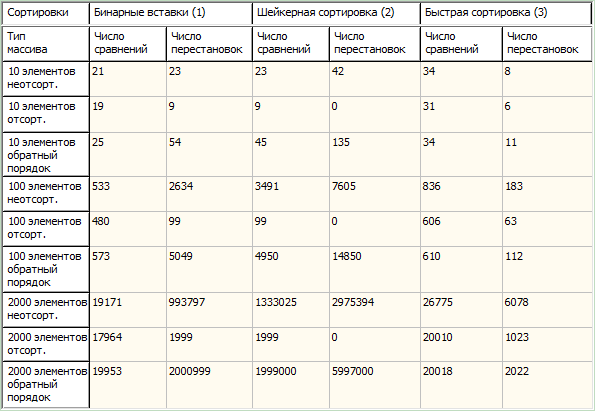


1. Быстрая сортировка:



1. **Таблица с результатами сортировок и расчётные значения**

Таблица с получившимися в результате работы программы значениями:



Расчётные значения:

1. Сортировка бинарные вставки:

Количество сравнений по формулам:

а) 10 элементов:

* avg: 23;

б) 100 элементов:

* avg: 460;

в) 2000 элементов:

- avg: 15897;

Количество перестановок по формулам:

а) 10 элементов:

– min: 9;

* avg: 31,5;
* max: 45;

в) 2000 элементов:

– min: 1999;

– avg: 1001499;

– max: 1999000;

б) 100 элементов:

– min: 99;

* avg: 2574;
* max: 4950;
* Значения приблизительно совпадают со значениями в таблице.

1. Шейкерная сортировка:

Количество сравнений по формулам:

а) 10 элементов:

– min: 9;

б) 100 элементов:

– min: 99;

в) 2000 элементов:

- min: 1999

Количество перестановок по формулам:

а) 10 элементов:

– min: 0;

* avg: 135;
* max: 67,5;

в) 2000 элементов:

– min: 0;

– avg: 5997000;

– max: 2998500;

б) 100 элементов:

– min: 0;

* avg: 14850;
* max: 7425;
* Значения приблизительно совпадают со значениями в таблице.

1. Быстрая сортировка:

Число сравнений при быстрой сортировке в обычном случае равно (n\*log2n) и может ухудшаться до n2 при самом плохом раскладе.

Количество сравнений по формуле (n\*log2n):

* 10 элементов – 33;
* 100 элементов – 664;
* 2000 элементов – 21932;

Значения приблизительно совпадают со значениями в таблице.

1. **Вывод об эффективности сортировок**
2. Сортировка бинарные вставки показывает эффективность работы только в отношении количества сравнений, количество перестановок имеет порядок 2, что является неэффективным, особенно для большого числа элементов.



1. Но по количеству перестановок QSort стабильнее выдаёт более маленькое их количество, чем другие перестановки. (Для подсчёта перестановок формул не приводилось)

Следовательно можно сделать вывод, что наиболее оптимальной сортировкой из трёх рассмотренных, если брать во внимание только количества сравнений и перестановок, является быстрая сортировка. И самой неэффективной является шейкерная сортировка.